Web Services Overview

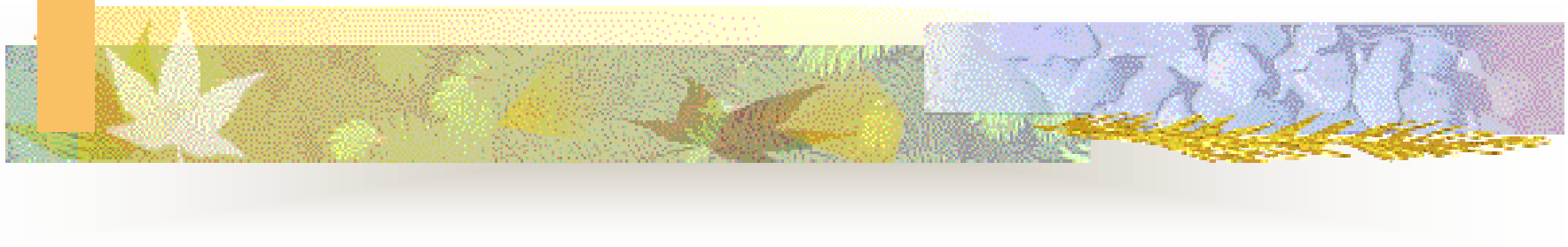
Francisco Curbera

IBM T.J. Watson Research Center

Outline

1. Why Web Services?
2. The Web Services Computing Stack.
3. Summary.

1. Why Web Services?



Today's Web

- Web designed for application to human interactions
- Served very well its purpose:
  - Information sharing: a distributed content library.
  - Enabled B2C e-commerce.
  - Non-automated B2B interactions.
- How did it happen?
  - Built on very few standards: http + html
  - Shallow interaction model: very few assumptions made about computing platforms.
  - Result was ubiquity.

What's next?

- The Web is everywhere. There is a lot more we can do!
  - E-marketplaces.
  - Open, automated B2B e-commerce.
  - Business process integration on the Web.
  - Resource sharing, distributed computing.
- Current approach is *ad-hoc* on top of existing standards.
  - e.g., application-to-application interactions with HTML forms.
- Goal:

  enabling systematic application-to-application interaction on the Web.

Web Services

"Web services" is an effort to build a distributed computing platform for the Web.

Yet another one!

Designing Web Services I

- Goals
  - Enable universal interoperability.
  - Widespread adoption, ubiquity: fast!
    - Compare with the good but still limited adoption of the OMG's OMA.
  - Enable (Internet scale) dynamic binding.
    - Support a service oriented architecture (SOA).
  - Efficiently support both open (Web) and more constrained environments.

Designing Web Services II

- Requirements
  - Based on standards. Pervasive support is critical.
  - Minimal amount of required infrastructure is assumed.
    - Only a minimal set of standards must be implemented.
  - Very low level of application integration is expected.
    - But may be increased in a flexible way.
  - Focuses on messages and documents, not on APIs.

Web Services Model

Web service applications are encapsulated, loosely coupled Web "components" that can bind dynamically to each other

2. The Web Services Framework

Web Services Framework

- Framework can be described in terms of
  - What goes “on the wire”:
    Formats and protocols.
  - What describes what goes on the wire:
    Description languages.
  - What allows us to find these descriptions:
    Discovery of services.

XML Messaging: SOAP

- SOAP 1.1 defined:
  - An XML envelope for XML messaging,
    - Headers + body
  - An HTTP binding for SOAP messaging.
    - SOAP is “transport independent”.
  - A convention for doing RPC.
  - An XML serialization format for structured data
- SOAP Attachments adds
  - How to carry and reference data attachments using in a MIME envelope and a SOAP envelope.

The SOAP Envelope

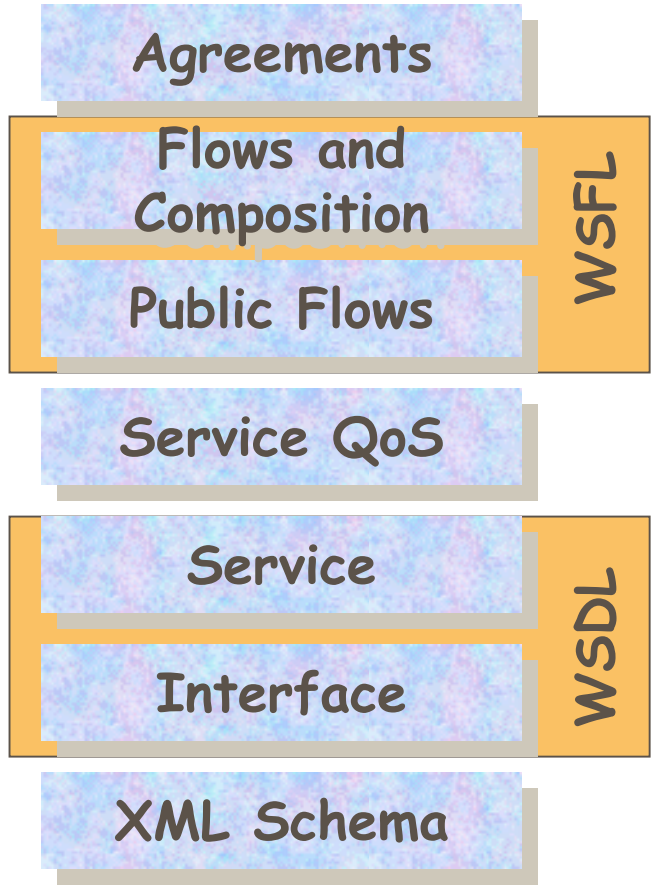
```
<SOAP-ENV:Envelope
  xmlns="http://schemas.xmlsoap.org/soap/envelope/">

    < SOAP-ENV:Header>
      ...
    </ SOAP-ENV:Header>

    < SOAP-ENV:Body>
       ...
    </ SOAP-ENV:Body>
   ...
</ SOAP-ENV: Envelope>
```

What goes on the wire

- Internet-scale integration needs a *lingua-franca*
  - XML messaging protocol over HTTP: SOAP
- Intra-enterprise integration needs to allow alternates:
  - CORBA, RMI
  - Messaging
  - In-memory method calls

Descriptions: Meta-data

- Integration requires interoperable machine-understandable descriptions
- Enables dynamic, delayed binding of components.
- Language extensibility provides support for different levels of application integration.

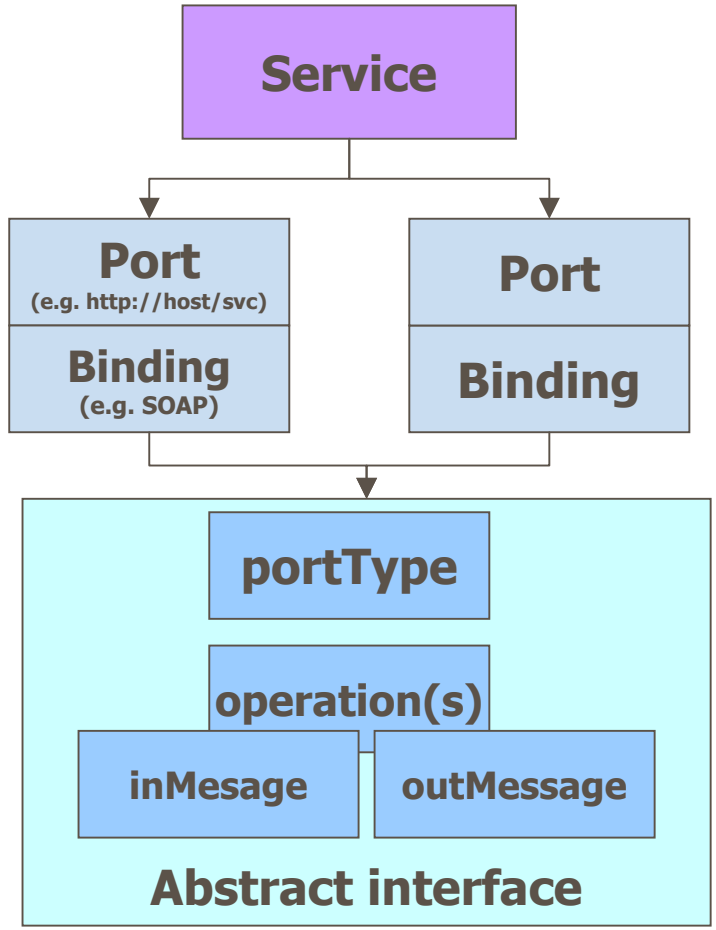


Web Services Description Language

- Provides functional description of network services:
  - IDL description
  - Protocol and deployment details
  - Platform independent description.
  - Extensible language.
- A short history:
  - WSDL v1.0, 9/2000
  - WSDL v1.1 submitted to W3C 3/2001.
  - A *de facto* industry standard.

WSDL Structure

- portType
  - Abstract definition of a service (set of operations)
- Multiple bindings per portType:
  - How to access it
  - SOAP, JMS, direct call
- Ports
  - Where to access it



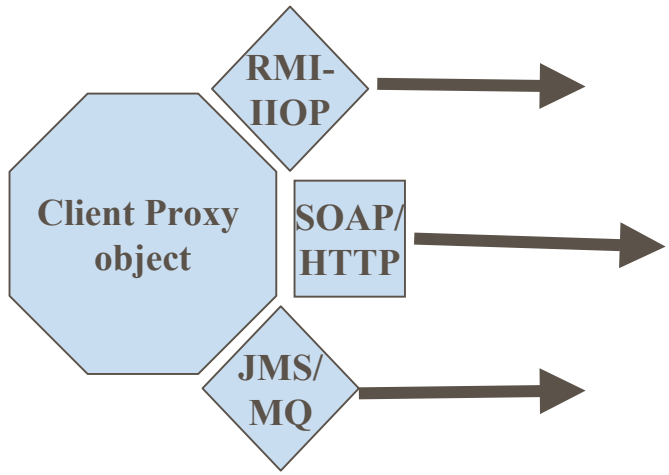
Using WSDL

1. As extended IDL: WSDL allows tools to generate compatible client and server stubs.
   - Tool support for top-down, bottom-up and “meet in the middle” development.
2. Allows industries to define standardized service interfaces.
3. Allows advertisement of service descriptions, enables dynamic discovery and binding of compatible services.
   - Used in conjunction with UDDI registry
4. Provides a normalized description of heterogeneous applications.

Client invocation

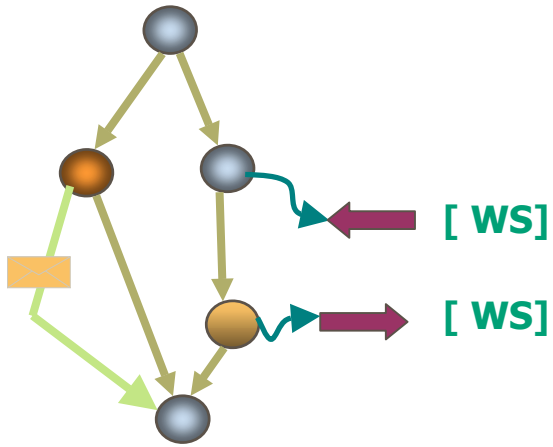
- Single stub can invoke services over different bindings
  - Depends only on abstract interface.
- Are independent of binding (but pluggable).
  - Add new bindings without recompiling/redeploying stub

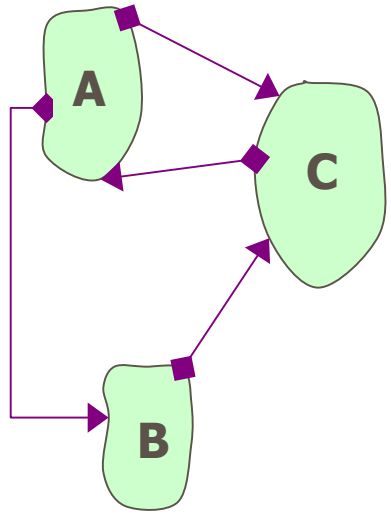
- Allows optimisations based on the bindings of service.
- Will support extended services models if described In WSDL



WSFL Overview

- WSFL describes Web Service compositions.
  1. Usage patterns of Web Services: describes *workflow or business processes*.
  2. Interaction patterns: describes *overall partner interactions*.

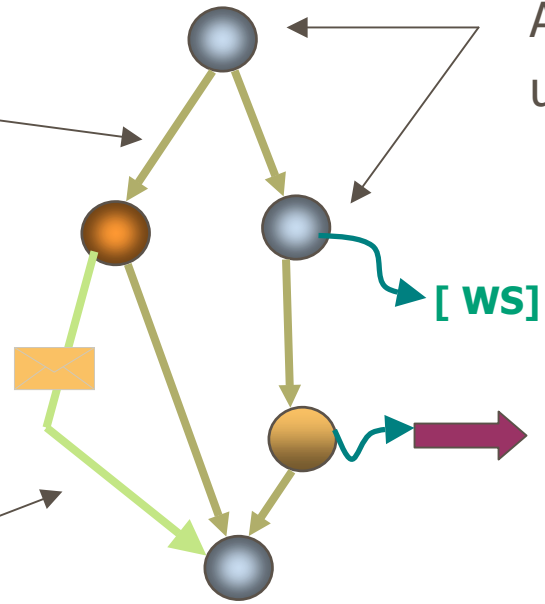




WSFL Flow Models

Control links define execution flow as a directed acyclic graph

Activities represent units of processing.



Activities are associated with specific typed **service providers**

Flow of data is modeled through data links.

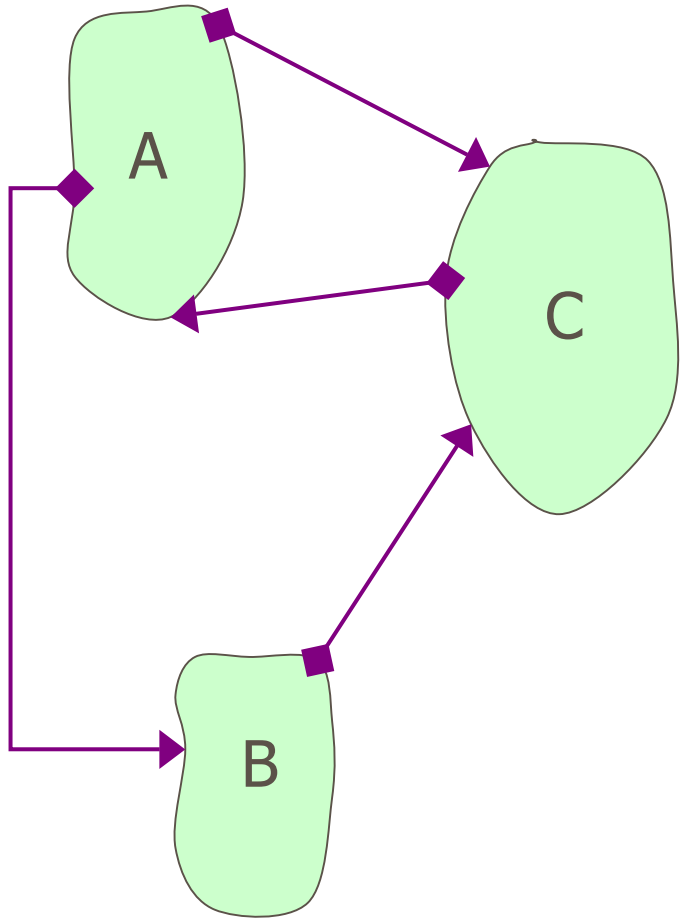
Activities can be mapped to the flow interface

Using Flow Models

- "Public flows" provide a representation of the service behavior as required by its users.
  - Typically, an abstraction of the actual flow begin executed
  - Defines a "behavioral contract" for the service.
  - Internal implementation need not be flow-based.
  - Flows are reusable: specify components types, but not what specific services should be used!
- "Private flows" are the flows executed in practice.
  - WSFL serves as a "portable flow implementation language"
- Same language is used in WSFL to represent both types of processes.

Global Models

- Global models describe how the composed Web Services interact.
  - RosettaNet automated.
  - Like an ADL.
- Interactions are modeled as links between endpoints of two service interfaces (WSDL operations).
- An essentially distributed description of the interaction.



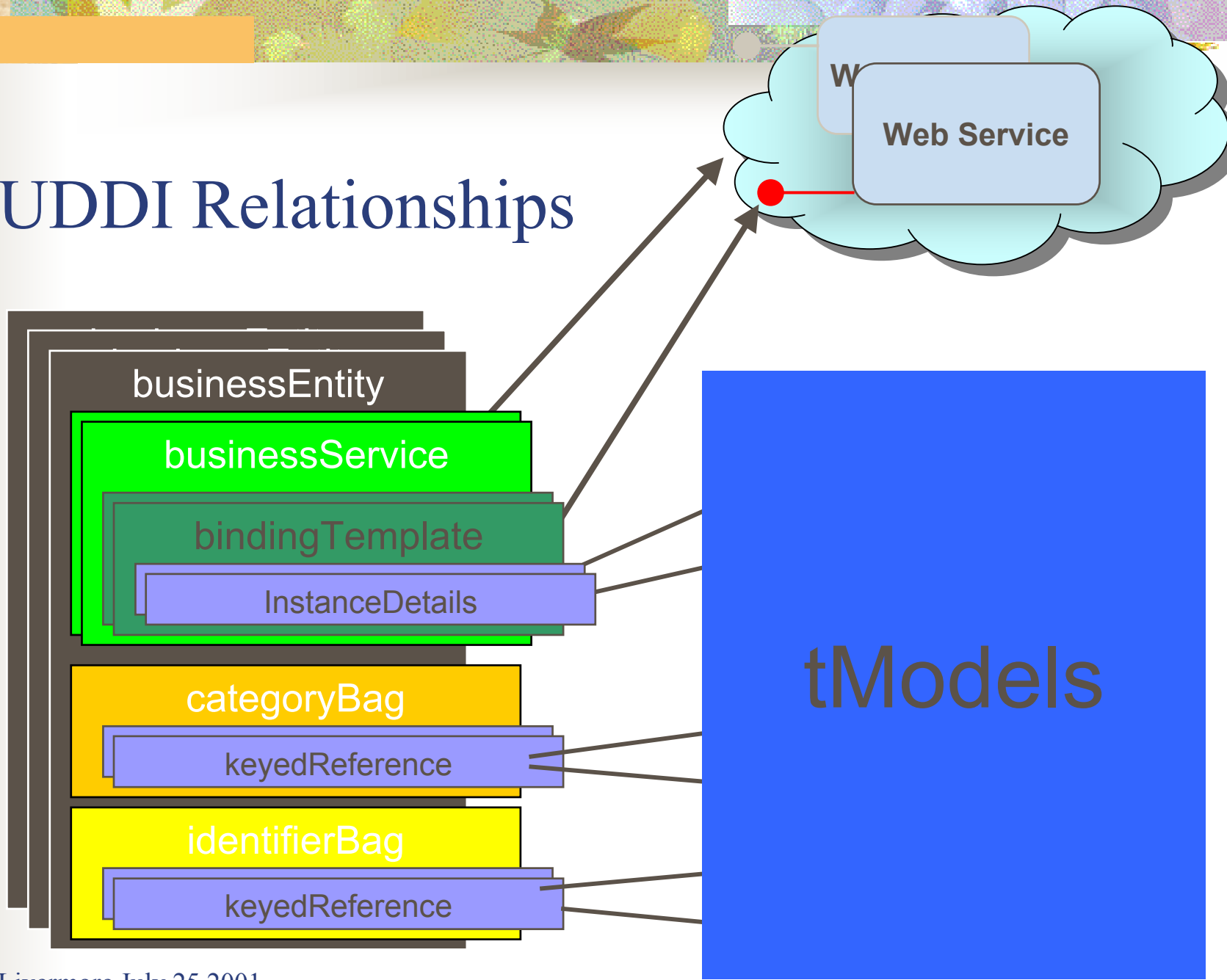
Discovery: Finding Meta-data

- Static binding requires service “libraries”.
- Dynamic binding requires runtime discovery of meta-data

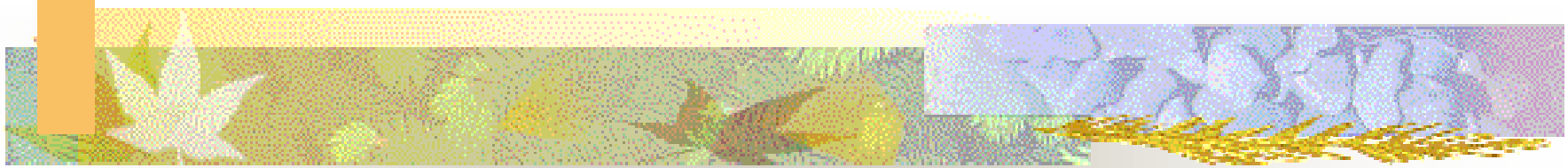
UDDI Overview

- UDDI defines the operation of a service registry:
  - Data structures for registering
    - Businesses
    - Technical specifications: tModel is a keyed reference to a technical specification.
    - Service and service endpoints: referencing the supported tModels
  - SOAP Access API
  - Rules for the operation of a global registry
    - “private” UDDI nodes are likely to appear, though.

UDDI Relationships



3. Summary



Summary

- The Web services framework is being defined, standardized and supported by the industry at a record pace.
- Broad industry acceptance and standard compliance will make it ubiquitous.
- Will bring an unprecedented level of interoperability to Web applications.
- The benefits of Web services, however, are not limited to the Web!

For more information

- SOAP

http://www.w3c.org/TR/soap

- WSDL

http://www.w3c.org/TR/wsdl

- UDDI

http://www.uddi.org

- WSFL

http://www.ibm.com/software/webservices

- Me:

mailto:curbera@us.ibm.com